Technická univerzita v Liberci
Hospodářská fakulta

BAKALÁŘSKÁ PRÁCE

2009 Jan Pavera

Technická univerzita v Liberci
Hospodářská fakulta

Studijní program: B 6209 Systémové inženýrství a informatika
Studijní obor: Podnikatelská informatika

Analýza nástrojů pro efektivní vývoj webových aplikací

Analysis of tools for essential progress of web applications

BP-PI-KIN-2009-19

JAN PAVERA

Vedoucí práce: Mgr. Tomáš Žižka, KIN

Konzultant: Petr Soukup, MITON CZ, s.r.o.

Počet stran: 31 Počet příloh: 3

22. května 2009

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladu, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci 22. května 2009

Poděkování

Chtěl bych poděkovat společnosti MITON CZ, s.r.o. za umožnění dlouhodobě řízené odborné praxe, za poskytnutí mnoha cenných zkušeností a informací nezbytných k vypracování mé bakalářské práce. Dále bych rád poděkoval vedoucímu mé práce Mgr. Tomáši Žižkovi za odborné vedení a správné nasměrování. Mé díky patří také garantovi mé praxe Petru Soukupovi, za trpělivost, vstřícný, ale profesionální přístup a mnoho užitečných rad. V závěru bych rád zmínil studenta VŠE oboru aplikovaná informatika Hynka Kvapila, který mi byl v mnoha problémech mé bakalářské práce oporou, tudíž mé díky patří i jemu.

Anotace a klíčová slova

V této práci bych rád popsal různé stupně využití programovacího jazyka PHP jako nástroje pro tvorbu webových aplikací a internetových stránek. Převážně vycházím ze zkušeností nasbíraných při mém působení na dlouhodobě řízené odborné praxi v internetové společnosti MITON CZ s.r.o. . Práce se skládá z čtyř částí. První část obsahuje základní popis webové aplikace a zaměřuje se na samotný programovací jazyk PHP, jeho historii, metodiku objektově orientovaného programování a porovnání verze PHP 4 s PHP 5. Druhá část popisuje účel a architekturu softwarové struktury framework, framework implementovaný na programovací jazyk PHP, příklady PHP frameworků a návrhový vzor MVC, na kterém je většina PHP frameworků postavena. Ve třetí části je rozebrán konkrétní PHP framework, ZEND Framework. Je zde popsána jeho struktura a implementace návrhového vzoru MVC. V poslední části je základní popis nástroje CMS, konkrétní příklad interního redakčního systému Dalén firmy Miton, který je na ZEND Frameworku založen a veřejného open source CMS Joomla!. V závěru bych rád zhodnotil jednotlivé úrovně nástrojů a jejich využití při vývoji webových aplikací.

Klíčová slova

Internet
Webová aplikace
PHP
PEAR
Objekt
Framework
MVC
Zend
CMS
Dalén
Joomla!

Annotation and key words

In this work, I would like to describe the various levels of use PHP programming language as a tool for creating web applications and websites. Largely based on the experiences collected in my action on long-term management of professional practice in internet company MITON CZ ltd.. The work consists of five parts. The first part contains the basic description of web applications and focuses on the PHP programming language itself, its history, object-oriented programming methodology and compare versions of PHP 4 to PHP 5. The second part describes the purpose and structure of the software architecture framework, framework implemented in the PHP programming language, examples of PHP frameworks and MVC design pattern, which is mostly PHP frameworks built. In the third part is dismantled concrete PHP framework, Zend Framework. This part is described its structure and implementation of MVC design pattern. In the last part is a basic description of CMS tool and, concrete example of internal content management system Dalén the project of Miton company, which is based on Zend Framework and public open source CMS Joomla! In conclusion I would like to evaluate the level of individual instruments and their use in developing of web applications.

Key words

Internet
Web application
PHP
PEAR
Object
Framework
MVC
Zend
CMS
Dalén
Joomla!

Obsah

Seznam pojmů a zkratek

W3C - World Wide Web Consortium, je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web.

HTML - HyperText Markup Language, značkovací jazyk pro vytváření stránek v systému World Wide Web.

CSS - Cascading Style Sheets, jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

XHTML - Extensible HyperText Markup Language, rozšiřitelný hypertextový značkovací jazyk je převedením jazyka HTML tak, aby vyhovoval podmínkám tvorby XML dokumentů a přitom byla zachována zpětná kompatibilita.

XML - eXtensible Markup Language, rozšiřitelný značkovací jazyk vyvinutý a standardizovaný konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Především je určen pro výměnu dat mezi aplikacemi a pro publikování dokumentů.

WML - Wireless Markup Language, značkovací jazyk založený na jazyce XML umožňující tvorbu online dokumentů pro mobilní zařízení.

Script - zdrojový kód programu, který je interpretován a spouštěn za běhu speciálním procesem, tzv. interpretem.

CGI - Common Gateway Interface**,** protokol pro propojení externích aplikací s webovým serverem.

Multiparadigmatický programovací jazyk – jazyk, který podporuje více než jedno programovací paradigma. Myšlenka multiparadigmatického programovacího jazyka je poskytnout framework, ve kterém mohou programátoři pracovat více styly, volně slučovat konstrukty rozdílných paradigmat. Hlavním cílem je poskytnout programátorům nejlepší nástroje k práci, připustíme-li, že žádné paradigma neřeší všechny problémy tou nejsnadnější a nejefektivnější cestou.

C - programovací jazyk, který vyvinuli Ken Thompson a Dennis Ritchie pro potřeby operačního systému Unix. V současné době je to jeden z nejpoužívanějších jazyků pro psaní systémového softwaru, ale je velmi rozšířený i pro aplikace.

Perl - je interpretovaný programovací jazyk vytvořený Larry Wallem v roce 1987. S rozvojem internetu se Perl stal velmi populárním nástrojem pro tvorbu CGI skriptů.

Python - interpretovaný objektově orientovaný programovací jazyk, který v roce 1990 navrhl Guido van Rossum.

Pascal - programovací jazyk, určený hlavně k výuce programování. Návrh pochází ze začátku 70. let od profesora Niklause Wirtha z Vysoké školy technické v Curychu. Jazyk Pascal sledoval tyto cíle: vhodnost pro výuku programování omezená počtem srozumitelných konstrukcí a navržení struktury jazyka tak, aby jeho implementace byla snadná na většině tehdejších počítačů.

Java - objektově orientovaný programovací jazyk, který vyvinutý firmou Sun Microsystems v roce 1995.

SQL - Structured Query Language, standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.

Oracle - relační databázový systém, vyvinutý firmou Oracle Corporation.

PostgreSQL - relační databázový systém vyvíjený skupinou PostgreSQL Global Development Group.

MySQL - relační databázový systém, vyvinutý švédskou firmou MySQL AB.

SQLite - relační databázový systém obsažený v relativně malé knihovně napsané v C.

Databázové transakce - skupina příkazů, které převedou databázi z jednoho konzistentního stavu do druhého.

Poddotazy - takový dotaz na databázi, který je umístěn uvnitř jiného vnějšího dotazu a výsledky z něj se používají v určité podmínce v tom vnějším dotazu.

Trigger - definuje činnosti databáze, které se mají provést v případě definované události nad databázovou tabulkou.

URL - Uniform Resource Locator, řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu.

HTTP - Hypertext Transfer Protocol, internetový protokol určený pro výměnu hypertextových dokumentů a souborů. Pomocí aplikačních bran zpřístupňuje i další protokoly.

FTP - File Transfer Protocol, protokol určený pro přenos souborů mezi počítači.

SMTP - Simple Mail Transfer Protocol, internetový protokol určený pro přenos zpráv elektronické pošty pomocí přímého spojení mezi odesílatelem a adresátem.

POP3 - Post Office Protocol version 3, internetový protokol, který se používá pro stahování e-mailových zpráv ze vzdáleného serveru na klienta.

IMAP - Internet Message Access Protocol, internetový protokol pro trvalý online vzdálený přístup k e-mailové schránce.

JavaScript - multiplatformní, objektově orientovaný skriptovací jazyk, fungující na straně klienta. Jeho autorem je Brendan Eich z tehdejší společnosti Netscape.

SSJS - Server-side JavaScript, JavaScript běžící na straně serveru

JSON - JavaScript Object Notation, je způsob zápisu dat nezávislý na počítačové platformě, organizována v polích nebo agregována objektech.

ASP - Active Server Pages, skriptovací platforma společnosti Microsoft, primárně určená pro dynamické zpracování webových stránek na straně serveru.

API - application programming interface, rozhraní pro programování aplikací je sbírka procedur, funkcí či tříd nějaké knihovny.

MVC - Model-view-controller softwarová architektura.

IDE - Integrated development environment, vývojové prostředí většinou zaměřené na jeden konkrétní programovací jazyk.

PHPUnit - unitový testovací framework pro přezkoušení a validování vyvíjeného PHP.

PDO - PHP Data Objects, knihovna vymezující lehké a konzistentní rozhraní pro přístup k databázím v PHP.

SOAP - Simple Object Access Protocol, je protokol pro výměnu zpráv založených na jazyku XML.

XML-RPC - XML Remote procedure call, je protokol, s jehož pomocí lze velice jednoduše provádět vzdálené volání procedur.

REST - Representational state transfer**,** je styl softwarové architektury pro distribuování hypermediálních systému, jako je World Wide Web.

Smarty - komplexní šablonovací systém pro PHP.

Seznam obrázků

1. Úvod

Téma mé bakalářské práce se nazývá analýza nástrojů pro efektivní vývoj webových aplikací. Zmiňované téma je však celkem obsáhlý pojem a zpracování všech těchto nástrojů je nad rámec mé bakalářské práce. Proto bych rád blíže specifikoval cíl a účel mé práce, objasnil zpracování a určil čtenáře, kterému je tato práce určena a bude mu nápomocna.

V první řadě bych se měl zmínit o mém působení na dlouhodobě řízené odborné praxi ve firmě MITON CZ, s.r.o. na pozici začínajícího programátora webových aplikací a stránek a částečně také projektanta. Ze zkušeností a znalostí nabytých při této praxi bude vycházet má práce. Její cíl tedy bude zpracovaní nástroje pro tvorbu webových aplikaci a stránek, se kterým jsem v průběhu působení na odborné praxi setkal a s nimž jsem pracoval.

Účelem mé bakalářské práce je předpoklad, že nastíní začínajícím vývojářům, budoucím profesionálům, případně samoukům, využitelné způsoby tohoto nástroje pro tvorbu webových aplikací z různých hledisek. Důležitý přínos bude mít také při prohloubení a ucelení mých zkušeností, které jsem na své dlouhodobě řízené odborné praxi získal. Rád bych tedy konkrétně popsal způsoby využití programovacího jazyka PHP při vývoji webových aplikaci, což bych také rád považoval za druhý cíl mé práce.

Vytyčit si hranice zpracování se stalo asi nejobtížnější části celého rozplánování mé práce. Rozhodl jsem se analyzované využití tohoto nástroje popsat spíše po základní teoretické stránce, nežli rozsah práce rozvádět do přílišných detailů.

O čtenáři, pro kterého je tato práce určena, jsem si již trochu zmínil a rád bych ho v tomto posledním odstavci blíže popsal. Budu předpokládat, že čtenář již nabyl zkušeností s tvorbou statických internetových stránek použitím značkovacího jazyka HTML. Zajímá se o skriptování, objektově orientované programovaní a prácí s databázemi. Orientuje se také ve fungování internetu a jeho službách.

V mé bakalářské práci bych se chtěl hlavně zabývat tvorbou webových aplikací a stránek využitím programovacího jazyka PHP, PHP frameworky, konkrétně ZEND Frameworkem. Rád bych vysvětlil pojem CMS (content management system), popsal konrétní interní redakční systém Dalén firmy Miton, který je na ZEND Frameworku založen a veřejný open source redakční systém Joomla!, se kterým jsem zkoušel pracovat ve svém volném čase.

2. PHP a OOP

2.1. Webová aplikace

Webová aplikace je aplikace poskytovaná prostřednictvím webového serveru uživatelům internetu, nebo vnitropodnikové obdobě intranetu. Webové aplikace jsou populární a stále více se rozšiřují především z důvodu všudypřítomnosti webového prohlížeče, schopnosti aktualizovat a spravovat webové aplikace bez nutnosti šířit a instalovat software na potenciálně tisíce uživatelských počítačů a hlavně také díky přístupnosti běžných desktopových aplikací pouze pomoci webového prohlížeče.

Webové aplikace generují dynamicky sérii internetových stránek ve standardním formátu HTML nebo XHTML. K vývoji těchto dynamických stránek se používá několik programovacích jazyků, kterými jsou například PHP, ASP, Perl, JavaScript nebo Java. Tyto programovací jazyky jsou však využívány v řadě systému a nástrojů pro vývoj webových aplikaci, jakými jsou framework nebo CMS. Má práce je ovšem zaměřena právě na programovací jazyk PHP.

2.2. Základní specifikace PHP

PHP je multiparadigmatický skriptovací programovací jazyk určený především pro programování dynamických internetových stránek a webových aplikací. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTML či WML. PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka je inspirována několika programovacími jazyky, jako jsou Perl, C, Pascal a Java. PHP je nezávislý na platformě a skripty fungují na mnoha různých operačních systémech. Programovací jazyk podporuje mnoho knihoven a dokáže přistupovat k většině databázových systémů, jako jsou MySQL, Oracle, PostgreSQL. Spolupracuje také s celou řadou internetových protokolů – např. HTTP, SMTP, FTP, IMAP, POP3. Programovací jazyk PHP patří do takzvaného LAMP, což označuje sadu open-source softwaru

používaného jako platforma pro implementaci dynamických webových stránek. Patří sem operačním systémem Linux, webový servere Apache, databázový systém MySQL nebo PostgreSQL a programovací jazyk PHP, Perl nebo Python.

2.3. Historie PHP

2.3.1. Předchůdce jazyka PHP[8]

V roce 1990 byla ve výzkumném centru CERN poprvé spuštěna služba World Wide Web. Službu podporovaly a podporují dodnes tři základní technologie. První z nich je jazyk HTML, který sloužil k zápisu stránek. Druhou nezbytnou technologií je protokol HTTP, který zajišťuje přenos HTML stránek z WWW serveru do prohlížeče. Třetí technologií nezbytnou pro implementování služby WWW jsou jedinečné URL adresy, která slouží k vytváření odkazů na daný objekt. Spojení těchto tří technologií však umožňuje pouhé prohlížení elektronických dokumentům provázaných systémem odkazů, v podobě statických internetových stránek.

Se vznikem rozhraní CGI přišla i první možnost automatického generování stránek, které obsahují informace proměnlivé v čase. Protokol CGI sloužil k propojení externích aplikací s webovým serverem. To serveru umožňovalo delegovat požadavek od klienta na externí aplikaci, která dle požadavku vrátila výstup. Taková aplikace typicky zpracuje nějaký skript ve webové stránce a webovému serveru navrátí statickou stránku, která je následně poslána klientovi jako výstup jeho požadavku. CGI skripty však byly velmi pomalé a proto se v roce 1996-97 na trhu objevily nové, rychlejší technologie SSJS a ASP.

SSJS, dříve LiveWire byla serverová podoba původně jen klientského JavaScriptu. Jsou velice podobné CGI skriptům, ale s tím rozdílem, že jsou psány rovnou do HTML stránky a jsou mnohem rychlejší a jednodušší. Technologii SSJS vyvíjí autor JavaScriptu Netscape. ASP jsou obdobou SSJS, pouze jejím autorem je společnost Microsoft. SSJS i ASP mají však jeden společný rozdíl oproti systému PHP, jsou to komerční

produkty a jejich použití je svázáno s použitím WWW serveru dané firmy. Princip použití PHP je obdobný jako u SSJS a ASP, celý produkt je však šířen jako open-source software.

2.3.2. Vývoj jazyka PHP[1]

PHP je nástupcem staršího produktu PHP/FI, který vytvořil dánsko-grónský programátor Rasmus Lerdorf v roce 1995, jako jednoduchou sadu skriptů v jazyce Perl pro zpracování záznamů o přístupech k jeho webu. Tuto sadu nazval "Personal Home Page Tools". Z důvodu větší funkčnosti, napsal Lerdorf rozsáhlejší implementaci v jazyku C, která byla schopna komunikovat s databázemi a uživatelům umožňovala vyvíjet jednoduché dynamické aplikace pro web. Zdrojový kód byl volně přístupný pro veřejnost, která mohla opravovat chyby a vylepšovat kód. PHP/FI znamená Personal Home Page / Forms Interpreter, obsahovalo něco ze základní funkcionality PHP, jak ho známe dnes. Mělo proměnné perlovského typu, automatickou interpretaci formulářových proměnných a syntaxi vloženou do HTML. Syntaxe samotná byla podobná jazyku Perl, přestože mnohem omezenější, jednodušší a v něčem nekonzistentní. V roce 1997 proběhla druhá implementace PHP/FI 2.0 psaná v C, která do projektu přilákala několik tisíc uživatelů po celém světě. Po většinu svého života však PHP/FI 2.0 strávilo v betaverzích, až bylo listopadu 1997 oficiálně uvolněno a následně překonáno první alfaverzí PHP 3.0.

PHP 3.0 bylo první verzí, která se velmi blížila PHP známému dnes. Vytvořili ho dva izraelští vývojáři Andi Gutmans a Zeev Suraski v roce 1997 kompletním přepsáním PHP/FI 2.0. Ve spolupráci s Rasmusem Lerdorfem zahájili budování PHP 3.0 nad existující uživatelskou základnou PHP/FI a rozhodli se prohlásit PHP 3.0 za oficiálního nástupce PHP/FI 2.0, jehož vývoj byl v podstatě zastaven. U PHP 3.0 došlo k obrovské možnosti rozšíření. Poskytlo pevnou infrastrukturu pro různé databáze, protokoly a API koncovým uživatelům. Přilákalo mnoho vývojářů, kteří se připojili a vytvořili nové rozšiřující moduly, což vedlo k jeho obrovskému úspěchu. Dalším klíčovým prvkem byla mnohem silnější a konzistentnější syntaxe a podpora objektově orientované syntaxe. Nový jazyk byl uvolněn pod nazván "PHP", což je rekurzívní akronym - PHP: Hypertext Preprocessor. Po cca 9 měsících veřejného testování bylo v červnu 1998 PHP 3.0 oficiálně

uvolněno a v době svého vrcholu bylo nainstalováno přibližně na 10% všech WWW serverů světa.

Díky novým možnostem a podpoře široké škály databází a API od jiných tvůrců začali Andi Gutmans a Zeev Suraski pracovat na přespání jádra PHP. Cílem návrhu bylo zvýšit výkon pro složité aplikace a zlepšit modularitu kódové báze PHP. Nový engine, nazvaný "Zend Engine" (sestaven z jejich křestních jmen), úspěšně splnil cíle návrhu a byl uveden v polovině roku 1999. Andi Gutmans a Zeev Suraski v tomto roce založili také firmu Zend Technologies v Ramat Gan, v Izraeli. PHP 4.0, založené na Zend Engine 1.0 a doplněné širokou škálou nových prvků, bylo oficiálně uvolněno v květnu 2000. K podstatně zvýšenému výkonu této verze přispěly další klíčové prvky, jako je podpora pro mnoho WWW serverů, HTTP sessions, buffering výstupu, bezpečnější způsoby zpracování vstupů uživatele a mnoho nových jazykových konstruktů. K vývoji PHP se přidalo několik stovek tisíc vývojářů.

Dne 13. června 2004 byla představena verze PHP 5, která již stojí na novém Zend Engine 2.0. PHP 5 obsahuje nově vylepšenou podporu pro objektově orientované programování, PHP Data Objects extension, ta definuje lehké a konzistentní rozhraní pro napojení k databázím a nesčetné množství výkonových vylepšení. PHP 4 se již dále nevyvíjelo a pro tuto verzi se nevydaly ani žádné bezpečnostní aktualizace. V roce 2008 se stává PHP 5 jedinou stabilní verzí, která se stále vyvíjí. Plánuje se další verze PHP 6, která se bude zároveň vyvíjet i s verzí PHP 5.

2.4. Základní specifikace OOP[6]

2.4.1. Základní koncepce

Objektově orientované programování (OOP, Object-oriented programming) je metodika vývoje softwaru, kdy chápeme procesy jako entity, tato metodika je založená na následujících základní koncepci:

- **Objekty** - jednotlivé prvky modelované reality, jak data, tak související funkčnost, které jsou v programu seskupeny do entit. Objekty si pamatují svůj stav a navenek poskytují operace přístupné jako metody pro volání.

- **Abstrakce** - je zjednodušení složité reality modelováním tříd daného problému, která je postavena na nejvhodnější úrovni dědičnosti pro daný aspekt tohoto problému.

- **Zapouzdření** - zaručuje, že objekt nemůže přímo přistupovat k datum a funkcím jiných objektů, což by mohlo vést k nekonzistenci. Každý objekt navenek zpřístupňuje rozhraní, pomocí kterého se s objektem pracuje.

- **Skládání** - objekt může obsahovat jiné objekty.

- **Delegování** - objekt může využívat služeb jiných objektů tak, že je požádá o provedení operace.

- **Dědičnost** - objekty jsou organizovány stromovým způsobem, kdy objekty nějakého druhu mohou dědit z jiného druhu objektů, čímž přebírají jejich schopnosti, ke kterým pouze přidávají svoje vlastní rozšíření. Tato myšlenka se obvykle implementuje pomocí rozdělení objektů do tříd, přičemž každý objekt je instancí nějaké třídy.

- **Polymorfismus** - je vlastnost, která umožňuje pojmenovat funkčnost objektu jedním jménem a tato funkčnost může být společná pro různé objekty ve stromové hierarchii, i když pro každý objekt v této hierarchii se bude chovat různě. Výsledkem je, že pro každý objekt ve stromové hierarchii bude mít volání funkčnosti jinou odezvu, avšak pro každý objekt vždy tu správnou.

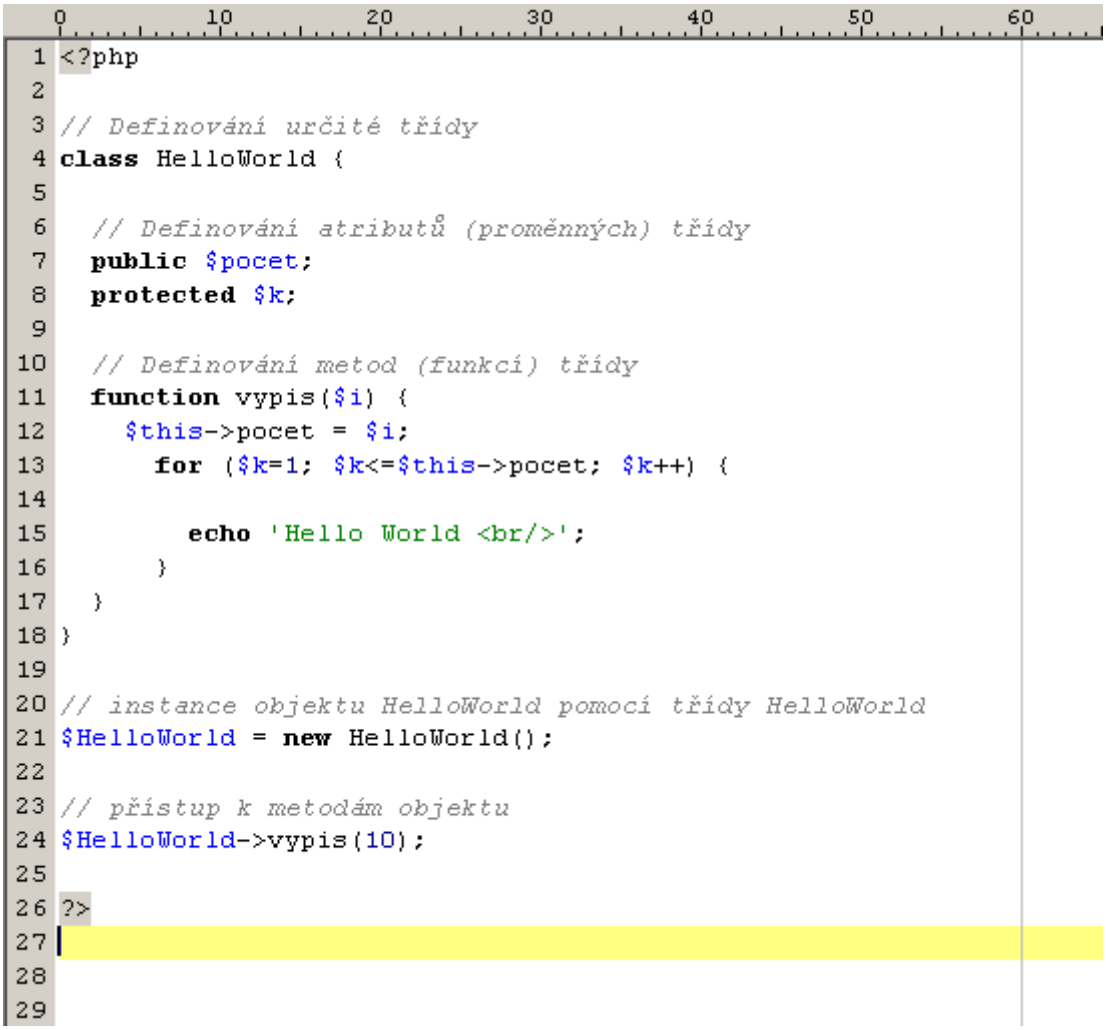
Výhody OOP se projeví zejména při vývoji rozsáhlých projektů, při kterých je nutné vytvořit velké množství zdrojového kódu. Na takových projektech většinou pracuje několik vývojářů. Vzhledem k tomu, že u OOP lze nastavit jasné rozhraní, je OOP ideálním prostředkem, jak danou aplikaci vyvíjet po více částech. Při použití OOP, nebude kód s nejvyšší pravděpodobností ani kratší a ani méně složitý. Často je tomu právě naopak, ale výhody spočívají v lepší správě a údržbě a také ve snadné rozšiřitelnosti a efektivní opakovatelnosti již jednou napsaných kódů.

2.4.2. OOP v PHP

V objektově orientovaném PHP tvoří objekt třída. Třída je šablonou objektu, volání třídy se pak označuje jako instance. Pojmy instance a objekt jsou v podstatě synonyma. Často je však konkrétní objekt považován jako instance, termín objekt se používá, pokud mluvíme o objektech obecně.

- **Třída** je to, co navrhujeme a programujeme.
- **Objekt** je to, co se z dané třídy vytváří za běhu aplikace.

Definici třídy tvoří atributy a metody. Atributy jsou specifické informace pro konkrétní instance, neboli proměnné reprezentující stav objektu. Metody jsou funkce s kterými objekt pracuje. Jednoduchý přiklad můžeme vidět v následujícím obrázku.



```php
<?php

// Definování určité třídy
class HelloWorld {

  // Definování atributů (proměnných) třídy
  public $pocet;
  protected $k;

  // Definování metod (funkcí) třídy
  function vypis($i) {
    $this->pocet = $i;
      for ($k=1; $k<=$this->pocet; $k++) {

        echo 'Hello World <br/>';
      }
  }
}

// instance objektu HelloWorld pomocí třídy HelloWorld
$HelloWorld = new HelloWorld();

// přístup k metodám objektu
$HelloWorld->vypis(10);

?>
```

Obr. 1 – příklad objektově orientovaného PHP

Zdroj:vlastní zpracování

Výstup tohoto PHP kódu vypadá takto:

Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

2.5. Verze PHP 4 a PHP 5

2.5.1. Porovnání verzí

Verze PHP 5 zlepšila PHP 4 zejména v těchto důležitých oblastech: objektově orientované programování (OOP), využití databází a XML. Verze PHP 5 má oproti PHP 4 nový objektový model. Manipulace PHP5 s objekty byla kompletně přepsána pro lepší výkonnost a více možností, které jsou dány hlavně těmito novými funkcemi.

- konstruktory
- destruktory
- public, protected a private proměnné a metody
- statické proměnné a metody
- konstanty v objektech
- abstraktní třídy
- interfaces – rozhraní
- přetěžování metod __get, __set, __call
- iterace přes datové členy
- metoda __toString pro převod do řetězce
- magické metody __sleep a __wakeup
- klíčové slovo final
- klonování objektů
- reflection API pro zpětnou analýzu objektů

A mnoho dalších. Významné změny se týkaly i databází. Hlavní změnou se stalo nové rozhraní pro MySQL nazvané MySQLi. Je to objektově orientované rozhraní, které využívá možností objektů v PHP 5, a které plně využívá možnosti MySQL verze 4.1. Velkou výhodou MySQLi (MySQLite) jsou zejména tyto podporované funkce.

- transakce
- poddotazy
- triggery
- a mnoho dalších pokročilých databázových funkcí

Od verze PHP 5 je opraven problém s XML rozšířením verze PHP 4, která na rozdíl od PHP 5 umožňovala manipulaci s XML pouze povrchně. Nové možnosti verze PHP 5 jsou obsaženy hlavně v těchto bodech.

- pracuje dohromady jako celek
- standardizováno na jedné XML knihovně libxml2
- plně vyhovuje specifikaci W3 konsorcia
- efektivně zpracovává data

Důležitou stránkou verze PHP 5 je díky propracovanější OOP efektivnější možnost API s kterou je spojen pojem framework. Příkladem takového dlouhodobě vyvíjeného frameworku pro PHP je PEAR. Frameworkům se budeme podrobněji věnovat v následující třetí kapitole.

2.5.2. PEAR

PEAR (PHP Extension and Application Repository) je distribučním systém knihoven a rozšíření pro aplikace psané v jazyce PHP, který vznikl z diskuse na PHP Developer' s Meeting v roce 2000. V současnosti archiv balíčků obsahuje více než 250 open source knihoven a rozšíření roztříděných tématicky do několika kategorií (např. databáze, souborový systém, HTTP, matematika, text, xml, web services apod.). Archiv doplňuje sofistikovaný distribuční systém, kvalitní dokumentace, návody a v neposlední řadě IRC kanál sloužící jak uživatelům, tak vývojářům nových projektů usilujících o začlenění do repositáře.

Jedním s "balíčků" PEAR je repositář PECL (PHP Extension Community Library), který obsahuje rozšíření pro samotné PHP. PECL má vlastní webové stránky a definuje vlastní styl psaní kódu, ovšem distribuční systém včetně struktury balíčků sdílí s PEAR. Mimo PECL jsou obzvláště důležité tzv. PHP Foundation Classes (PFC), což je další skupina balíčků, které standardní distribuce PHP instalují spolu s nástroji pro práci s archivem. PFC tvoří jen stabilní balíčky, které nejsou vázány na konkrétní systém a jsou díky standardizovanému API do budoucna snadno rozšiřitelné. Pokud budou distribuce PHP i nadále instalovat balíčky z PEARu, budou to ty z PFC.

2.5.3. Praxe - sázková aplikace

Verze PHP 4 se v dnešní době už nevyvíjí, ale samozřejmě je součástí nastupující verze PHP 5. Dle mého názoru usuzuji, že by si každý začínající vývojář webových aplikací v tomto jazyce měl projít menším projektem vytvořeným právě "neobjektové" strukturované verzi PHP 4, pokud dobře nezná OOP. Konkrétním příkladem muže být můj první úkol na dlouhodobě řízené odborné praxi ve firmě Miton.

2.5.3.1. Úkol

Úkolem tohoto zadání bylo vytvořit dynamickou webovou sázkovou aplikaci pomocí skriptovacího programovacího jazyka PHP 4 a databázového systému MySQL. Na tomto zkušebním projektu jsem pracoval s kolegou Janem Mrázkem. Základem aplikace byla tato funkcionalita: registrace nového uživatele, zalogování a odlogování uživatele, možnost úpravy profilu uživatele, rozhraní pro zadávaní sazek podle určitého hracího kola, výpis zápasu pro určité hrací kolo, výpis zadaných sázek a výpočet možné výhry, administrační rozhraní pro vkládaní zápasu, kurzu, aktuálního hracího kola a možnost výpisu uživatelů. Veškerá vstupní a výstupní data měla být uložena v námi efektivně navržených tabulkách databáze.

2.5.3.2. Přínos

Nejdůležitějším přínosem při vývoji sázecí aplikace pro mě bylo seznámení se základy jazyka PHP a MySQL. Pochopení struktury odkazů, práce s HTML formuláři, globálními proměnnými a funkcemi. Dodržení správné syntaxe. Propojení PHP a MySQL. Použití jazyka DDL a DML při práci s databázi.

2.5.4. OOP nebo strukturované programování?

V programovacím jazyku PHP jde vytvořit všechno vcelku rychle, ať už je to rychlost práce aplikace nebo její vývoj. Rychlý vývoj aplikací často znamená malou strukturu a několik krátkých spolu nesouvisejících skriptů. To nemusí ale být vždy nevýhodou. Právě drobný skript v PHP, který splní vše potřebné, může být tím nejideálnějším řešením. Naproti tomu však objektově orientované programování poskytuje jasnou strukturou, rozhraní a promyšlený koncept a proto je u rozsáhlých, datově náročných a profesionálních projektů v určitých případech jasně efektivnější.

3. PHP Frameworky

3.1. Základní specifikace Frameworku

3.1.1. Účel

Framework je softwarová struktura nebo také kostra, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Framework je množinou podpůrných programů, kódů nebo knihoven API, které poskytují běžně používanou funkcionalitu pro mnoho typů aplikací. Zatímco jedna knihovna obvykle nabízí jednu speciální funkcionalitu, tak framework poskytuje široké spektrum funkcionalit, které mohou být danou aplikací využívané.

Cílem je převzetí a zaměření na typické a stále se opakující problémy v dané oblasti. Diky předpřipravené množině souborů, které obsahují otestované struktury a souhrn znalostí zkušených vývojářů, framework usnadňuje a zrychluje vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání a zaměřit se na specifické požadavky klienta, pro které framework nemá připravené řešení, na místo opakovaného programování nejpoužívanějších a obecných funkcí daného projektu.

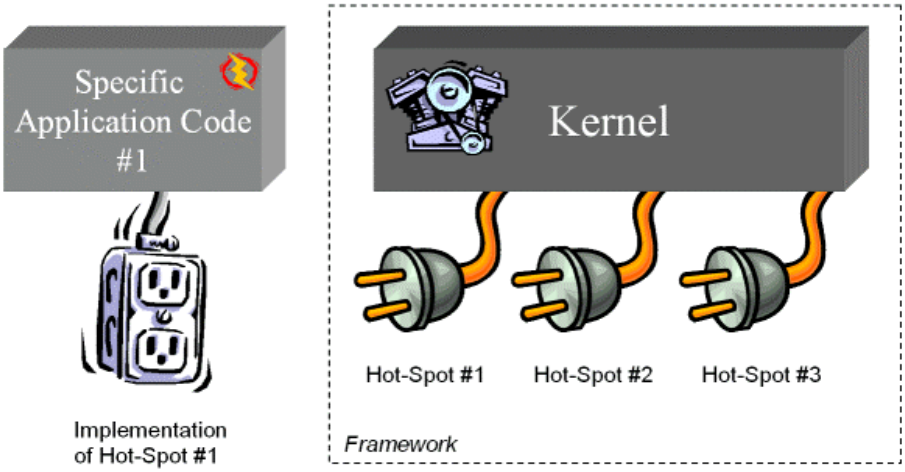
Vyskytují se námitky, že použitím frameworku bude kód pomalý či jinak neefektivní a že čas, který se ušetří použitím cizího kódu, se musí věnovat nastudování frameworku. Nicméně však při možnosti nasazení znovupoužitelných prvků a využití v rozsáhlém, na funkcionalitu náročném projektu dojde k výrazné úspoře času. Díky tomu mají frameworky velké zásluhy na zrychlení a zefektivnění vývoje webových aplikací.

3.1.2. Architektura[7]

Architektura frameworků se skládá z tzv. “frozen spots“ a “hot spots“. Frozen spots definují celkovou architekturu softwarové struktury, základní komponenty, jejich vztahy a tím tak tvoří jádro frameworku. Tyto části jsou neměnné, konstantní a vždy součástí každé instance frameworku. Naproti tomu hot spots jsou komponenty, které spolu s kódem programátora vytvářejí zcela specifickou funkcionalitu, a proto jsou skoro pokaždé jiné. Vycházejí ovšem z jádra frameworku.

V objektově orientovaném prostředí je framework tvořen abstraktními a klasickými (neabstraktními) třídami. Hot spots pak mohou být reprezentovány abstraktními třídami a vlastní kód se vytvoří implementací abstraktních metod.

Metaforicky by se to dalo přirovnat k motoru, který vyžaduje sílu, avšak na rozdíl od tradičního motoru, motory frameworku potřebuje více zdrojů síly. Každý z těchto motorů je hot spot frameworku a každý tento motor musí být napájen specifickou silou, aby mohl pracovat. Do každého hot spotu musí být tedy připojen specifický kód, frozen spot z jádra frameworku. Motor nebude fungovat, dokud nebudou všechny zásuvky připojeny.



Obr. 2 - Metafora frameworku

Zdroj: http://www.acm.org/crossroads/xrds7-4/frameworks.html

3.2. PHP Frameworky

V jazyce PHP je napsáno několik frameworků, které nám ulehčují práci při psaní webových aplikace a pracují s návrhovým vzorem MVC. Použití některého z nich umožní plné využití výhod MVC architektury a díky vestavěným funkcím také zajistí méně napsaného kódu a tím pádem rychlejší vývoj požadované aplikace.

3.2.1 MVC architektura[5]

MVC je architektonická struktura používána v softwarovém inženýrství, která při úspěšném použití návrhového vzoru izoluje část obchodní logiky od uživatelského rozhraní aplikace. Tato softwarová architektura rozděluje aplikaci na 3 logické komponenty tak, že modifikace některé z nich má minimální vliv na ostatní. Tyto komponenty jsou:

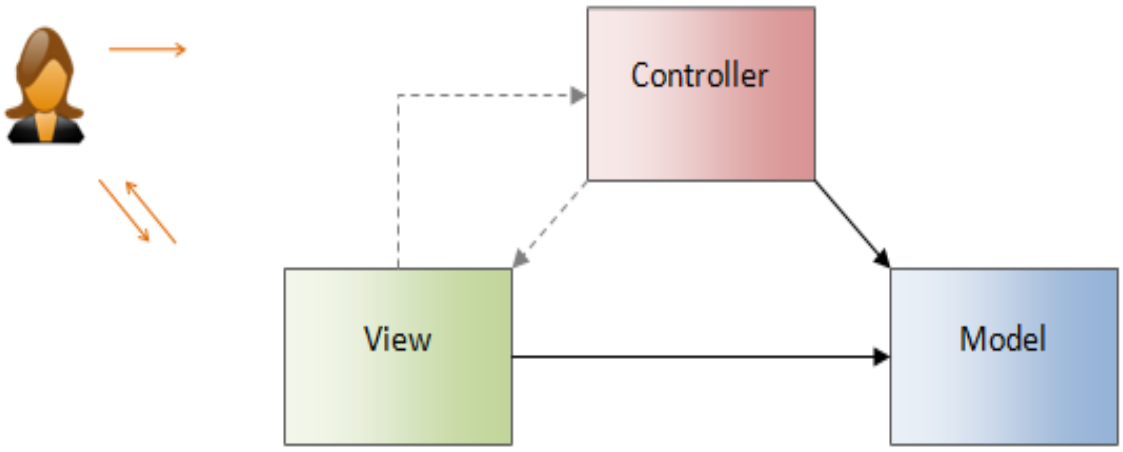
- **Model** - model
- **View** - pohled
- **Controller** - ovladač

Jednotlivé komponenty při využití Architektury MVC jako návrhového vzoru PHP frameworků:

Model - je datový model, neboli také doménově specifická reprezentace informací, s nimiž aplikace pracuje. Model je zodpovědný za výběr dat, která budou zobrazena.

View - vykreslí data reprezentovaná modelem do podoby vhodné k interaktivnímu uživatelskému rozhraní.

Controller - se stará o tok událostí v aplikaci a obecně aplikační logiku, reaguje tedy na události pocházející od uživatele a zajišťuje změny v modelu nebo v pohledu.



Obr. 3 - Schéma MVC

Zdroj: http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc

3.2.2. Příklady php frameworků[2]

Na první pohled se může zdát, že frameworků pro PHP je velké množství. Pokud se však na výběr zaměříme detailněji, zjistíme, že ne každý vyhovuje našim představám. Některé frameworky mají nepřeberné množství knihoven, které obstarávají téměř vše, na co si vzpomenete. To je ovšem vykoupeno jejich vyššími hardwarovými nároky a nižší rychlostí. Jiné mají knihoven méně, ale dají se díky své modulárnosti lehce rozšířit o nové funkce. Velkou pozornost musíme věnovat dokumentaci jednotlivých frameworků. Pokud je dokumentace na špatné úrovni, práce s frameworkem trvá mnohem déle, protože je třeba dlouze vyhledávat parametry funkcí či řešení jiných úskalí.

Více informací o PHP frameworkách poskytuje web www.phpframeworks.com, na kterém jsou jednotlivé frameworky porovnávány podle následujících parametru:

- **MVC** - podpora architektury Model-View-Controller.
- **Multiple DB's** - podpora více databází.
- **ORM** - podpora object-record mapper, obvykle implementací návrhového vzoru ActiveRecord.

- **DB Objects** - framework obsahuje další databázové objekty, jako TableGateWay.
- **Templates** - framework má vestavěné šablony.
- **Caching** - framework zahrnuje cachování objektu či nějaký jiný způsob cachování.
- **Validation** - vestavěné validování nebo filtrování component.
- **Ajax** - vestavěná podpora pro Ajax.
- **Auth Module** - vestavěný modul pro manipulaci s ověřením uživatele.
- **Modules** - framework obsahuje ostatní užitečné moduly, jako RSS feed parser, PDF modul atd.
- **EDP** - možnost Event Driven programování.

Příklady PHP frameworků nalezneme v příloze A - strana 49.

4. ZEND Framework

4.1. Základní specifikace

Zend Framework je PHP MVC framework vyvíjený od počátku roku 2005 společnosti Zend Technologies, která je tvůrcem nejpokročilejšího IDE pro PHP. V současné době se podílí na vývoji více než 300 vývojářů s širokou praxí a mnohaletými zkušenostmi s jinými frameworky jako je .NET nebo Ruby on Rails. Tyto zkušeností jsou maximálně vkládány do vývoje Zend Framworku. Mezi vývojáři a konzultanty patří například takové osobnosti, jako jsou Sebastian Bergmann (tvůrce PHP Unit), Andi Gutmans (tvůrce moderního PHP) či George Schlossnagle (autor jedné z nejlepších knih o pokročilém PHP). Přibližně 10 vývojářů jsou přímo zaměstnanci firmy Zend.

Dokumentace Zend Frameworku je velmi propracovaná a dobře se v ní orientuje. Na oficiálním webu www.framework.zend.com, kde jsou veškeré knihovny a komponenty zdokumentované, lze nalézt i různé video návody, prezentace a řešení určitých problémů. Nejen díky tomu je tento framework celosvětově velmi rozšířený a komunita kolem něj je obrovská. Zend má mimo jiné velké zastoupení i mezi českými vývojáři.

Firma MITON CZ, s.r.o. vyvíjí většinu svých projektů právě na technologii Zend Framework. Proto bych chtěl ve své bakalářské práci popsat právě tento framework a to konkrétně verzi 1.5.

4.2. Struktura

4.2.1. Knihovny (komponenty)[3]

- **Zend_Acl** - knihovna, která umožňuje spravovat přístup uživatelů k jednotlivým sekcím podle uživatelských práv. Můžeme tedy určit uživatelské role, jako je návštěvník, registrovaný, člen, administrátor a přidělit jim určitá práva. Tato knihovna rovněž umožňuje dědičnost práv. V konkrétním příkladu blogu by to například znamenalo, že návštěvník si může prohlížet články, registrovaný může navíc články komentovat, uživatel může provádět stejné operace jako registrovaný, ale navíc může články vydávat a administrátor má neomezené možnosti.

- **Zend_Auth** - knihovna Zend_Auth umožňuje přihlašování a odhlašování uživatelů a udržování informací v podobě sessions o tom, zda aplikace komunikuje se stále stejným uživatelem.

- **Zend_Cache** - kešování je součástí každé webové aplikace a právě s tím pracuje tato knihovna, která umožňuje kešovat výstup funkce, třídu, soubory a pro ukládání keše používá například databázi SQlite.

- **Zend_Config -** slouží k nastavení aplikace skrze konfigurační soubory INI nebo XML.

- **Zend_Console_Getopt** - zpracovává volby a argumenty z příkazového řádku.

- **Zend_Currency** - knihovna sloužící pro formátování a lokalizaci měny.

- **Zend_Controller -** nejdůležitější a velmi rozsáhlá knihovna Zend Frameworku, která pomocí návrhového vzoru FrontController vytváří MVC aplikace.

- **Zend_Date -** jednoduchá knihovna umožňující zobrazovat čas a datum ve více než sto formátech.

- **Zend_Db -** jedna z nejrozsáhlejších knihoven, která umožňuje pracovat se všemi databázemi prostřednictvím jednoho rozhraní. Pro přístup k databázi používá vrstvu PDO, ale i adaptér pro třídu mysqli. Knihovna obsahuje také parser, který přizpůsobí SQL dotaz pro konkrétní databázi a profiler, podle kterého lze zjišťovat, kolik času zabralo spojení s databází, kolik času se vykonávaly všechny dotazy pro výběr, změnu a podobně.

- **Zend_Debug -** knihovna, která pomoci funkce var_dump() efektivně vrací přehledně zformátované výstupy.

- **Zend_Exception -** knihovna se základní výjimkou, od které jsou odvozeny veškeré ostatní výjimky frameworku.

- **Zend_Feed** - obsahuje nástroje pro práci s Atom a RSS výstupy.

- **Zend_Filter** - knihovna pro filtrování vstupních uživatelských dat. Podporuje například blacklisting - odstraňuje škodlivá data, whitelisting - propouští pouze správná data, blind filtring - z dané hodnoty vybere jen to potřebné a jiné.

- **Zend_Form** – knihovna, která slouží pro zpracování formulářů. V aplikaci lze nadefinovat pravidla pro formulář, jakými jsou například povinné položky, filtry, validátory, pravidla pro zobrazení a jiné.

- **Zend_Gdata** - rozhraní pro práci se službami nabízenými společností Google (Base, Calendar, Blogger, CodeSearch).

- **Zend_Http** - knihovna, která umožňuje práci s HTTP protokolem, cookies, autentizaci přes HTTP, posílání souborů a podobně.

- **Zend_Json** - knihovna určena k práci s JSON objekty.

- **Zend_Layout** - knihovna efektivně řešící načítání hlavičky a patičky velmi snadnějším způsobem, než v předešlých verzích.

- **Zend_Loader** - poskytuje metody pro nahrávání tříd do aplikace.

- **Zend_Locale** - umožňuje lokalizaci různých formátů dat (datum, čas, měna a podobně), tím že propojuje knihovny Zend_Date, Zend_Measure, Zend_Translate, Zend_Currency a Zend_TimeSync.

- **Zend_Log** - knihovna s jednoduchým rozhraním pro ukládání informací do logu.

- **Zend_Mail** - knihovna pro snadnou práci s e-maily a protokoly POP3 nebo SMTP.

- **Zend_Measure** - knihovna umožňující konvertování hodnot různých jednotek míry.

- **Zend_Mime** - obsahuje nástroje pro práci s standardem MIME, knihovnu využívá hlavně Zend_Mail.

- **Zend_Pdf** - knihovna pro jednoduché rozhraní pro práci s PDF dokumenty.

- **Zend_Registry** - pracuje s globálními daty.

- **Zend_Rest, Zend_Soap, Zend_XmlRpc** - obsahuje třídy pro práci s REST a protokolem SOAP a XML_RPC.

- **Zend_Search** - knihovna umožňující prohledávat textové soubory.

- **Zend_Server_Reflection** - knihovna diky Reflection API pro PHP 5 získává různé informace o třídách a funkcích.

- **Zend_Service_*** - skupina komponent pracujících s API různých webových služeb. Například Akismet, Amazon, Audioscrobbler, Delicious, Flickr, Simpy, StrikeIron, Technorati, Yahoo a v budoucnu ještě mnoho dalších

- **Zend_Session** - knihovna obsahuje mechanizmy pro práci se sessions.

- **Zend_Uri** - slouží především pro ostatní knihovny k manipulování a validování URI.

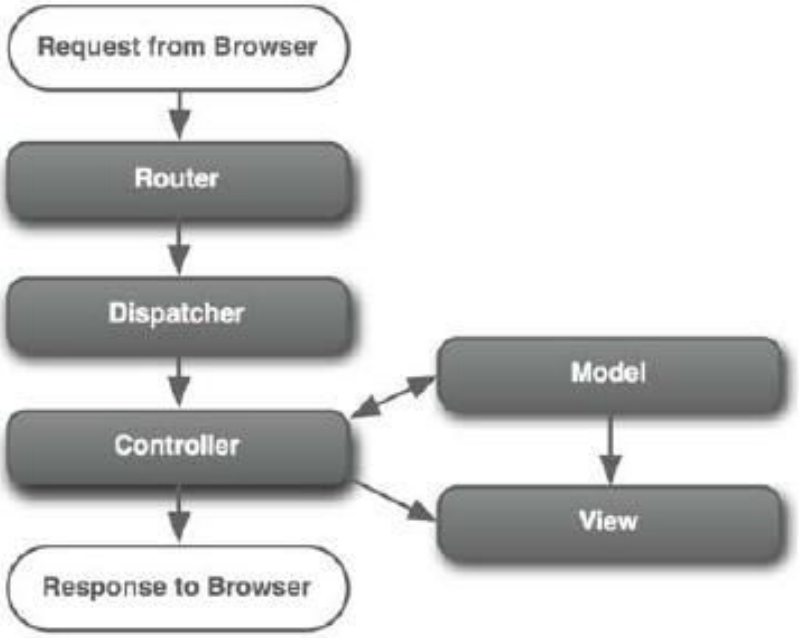
- **Zend_Translate** - umožňuje spravovat překlady v různých jazykových formátech.

- **Zend_Validate** - knihovna umožňující zautomatizovat validaci určitých dat.

- **Zend_Version** - vrací aktuální verzi Zend Frameworku.

- **Zend_View** – knihovna vytvářející výstupy z frameworku. Umožňuje integrovat různé šablonovací systémy, například Smarty.

4.2.2. MVC architektura[4]



Obr. 4 - Implementace MVC v Zend Frameworku

Zdroj: ALLEN ROB, LO NICK, STEVEN BROWN, Zend Framework in Action.

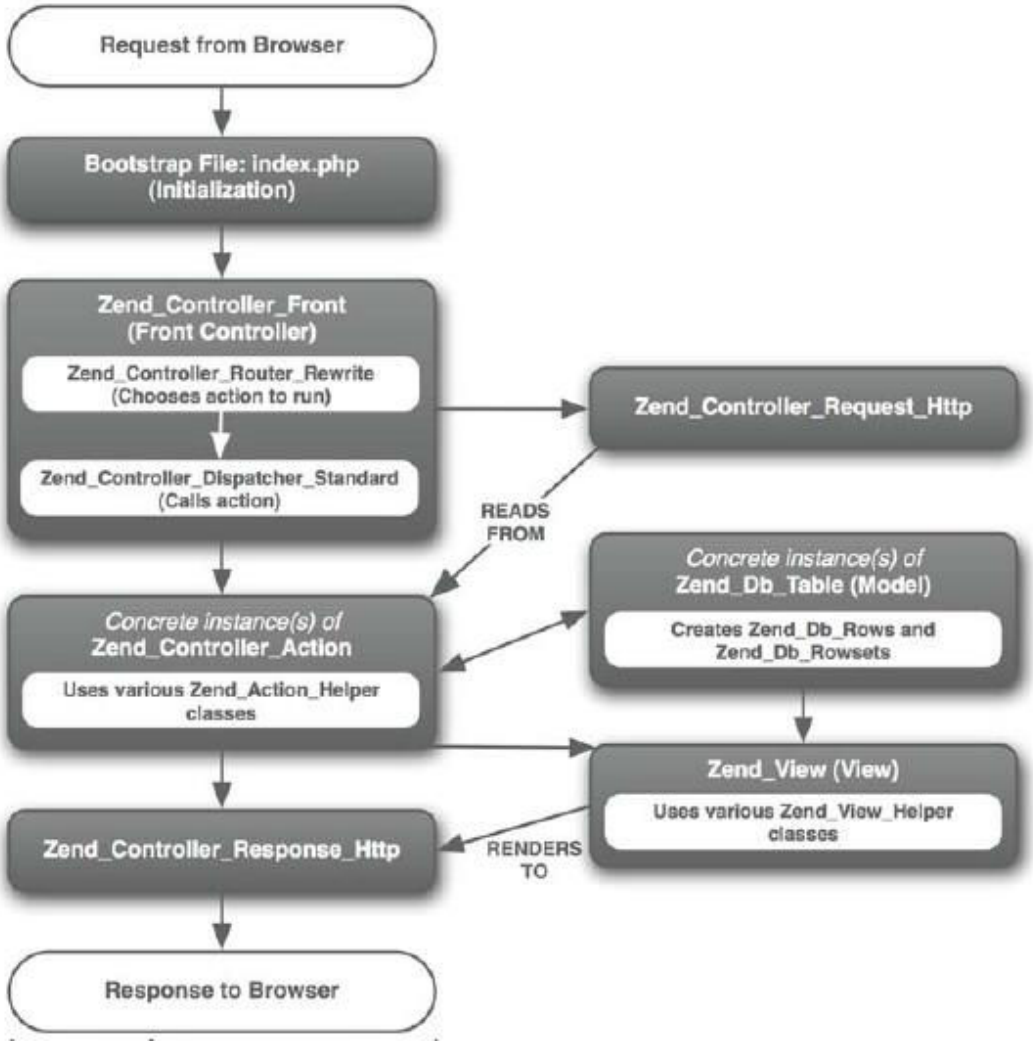
- **Request** - ukládá veškerá data získaná z URL ve výchozí podobě do objektu požadavku, který je předáván všem částem knihovny ovladače.

- **Router** - zajišťuje routing proces, při kterém se z URL získává název modulu, třídy, metody, parametrů a ukládá se do objektu požadavku.

- **Dispatcher** - provádí proces, který vytváří instance ovladače a volá jeho konkrétní metodu. Narozdíl od routingu se dispatching může provádět vícekrát.

- **Response** - odpovědi zajišťuje objekt, kde se ukládají veškeré inicializace a nastavení uvnitř volané metody.

Implementace návrhového vzoru MVC je v Zend Frameworku promítnuta do knihovny Zend_Controller. Hlavním úkolem této knihovny je rozpoznat z URL adresy moduly a třídy, které se budou při sestavování stránky používat. Zjednodušeně celý proces vypadá tak, že po vytvoření objektu třídy Zend_Controller_Front její parser rozpozná modul, třídu a metodu třídy. Tyto informace uloží do objektu požadavku, který je předán třídě a metodě rozpoznané parserem. Pak se metoda třídy zavolá, inicializuje veškerá potřebná data a předá je do objektu odpovědi, který je následně předán jako výsledek operace.



Obr. 5 - Zpracování operace v Zend Frameworku

Zdroj: ALLEN ROB, LO NICK, STEVEN BROWN, Zend Framework in Action.

5. CMS

5.1. CMS

CMS (content management system) je systém pro správu obsahu, někdy oborově nazývaný redakční nebo publikační systém. CMS je software zajišťující správu dokumentů webového obsahu, například obsahu internetových prezentací, internetových médií a jiných projektů. Hlavím cílem je oddělit role jednotlivých osob, které spolupracují při tvorbě a provozu webových aplikací, především tvůrce technické a grafické části, tedy programátory a grafiky od administrátorů nebo sitebuilderů.

V dnešní době jsou CMS zpravidla vyvíjeny jako webové aplikace, někdy s případnými doplňkovými programy u klienta. Člení se dle řady kritérií, například podle rozsahu řešení, použitého vývojového prostředí nebo cílové skupiny. Nejjednodušší CMS jsou programovány v jazyku JavaScript, řada CMS jsou ovšem vyvíjeny pomocí jazyka PHP a databázového systému MySQL, využíván je i jazyk Java a jiné programovací jazyky.

Trh s CMS programy je široký, existuje řada programů nabízených jako free software, kterým je například Joomla!, komerční produkt nebo jako interní firemní aplikace, kterou je právě redakční systém Dalén firmy Miton.

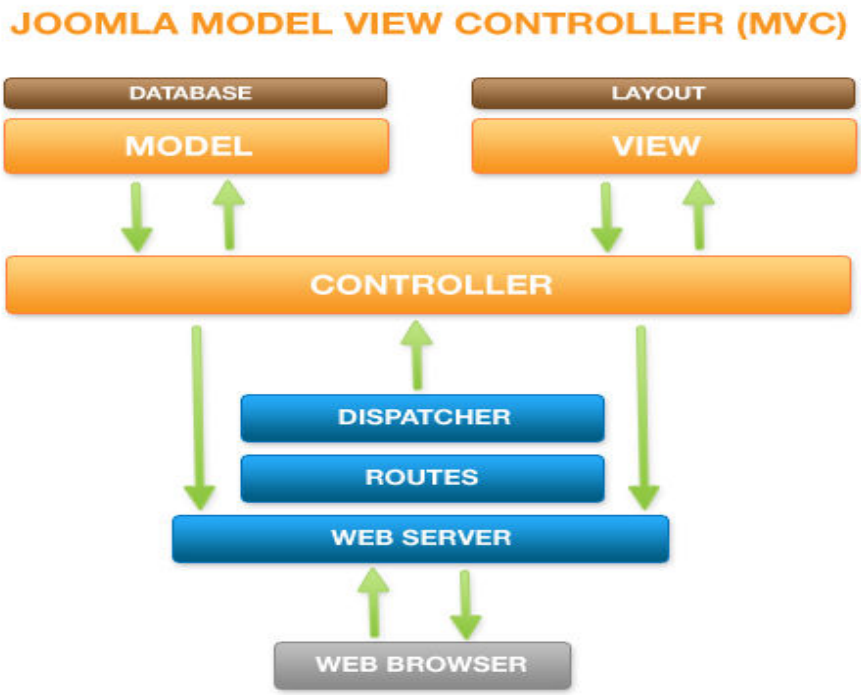
5.3. Základní popis Joomla! CMS

Joomla! CMS je bezplatný open source redakční systém pro účely publikování informací na internetu a intranetu. Podporuje mnoho komponent pro vytvoření vlastního obsahu webu. Komponenty například pro:

- caching
- indexaci stránek
- RSS
- tisknutelné verze stránek
- zobrazování novinek
- blogy, diskusní fóra
- hlasování
- kalendář
- vyhledávání v rámci webserveru
- online obchody
- řešení e-komerce
- systém reklamních bannerů
- multimediální galerie
- předplatné systémy
- správa dokumentů
- lokalizace a vícejazyčné verze.

Joomla! je ve své podstatě také aplikační framework postavený na architektuře MVC. Podporuje vytváření sofistikovaných doplňků a rozšíření, proto se na jeho vývoji podílí několik stovek tisíc profesionálních vývojářů z oficiálních a neoficiálních komunit po celém světě. Díky jednoduchému administračnímu prostředí je však vhodná i pro amatéry, kteří si mohou sestavit vlastní webovou aplikaci, tvořit si obsah, používat šablony a rozšiřovat Joomlu o nové komponenty se specifickými funkcemi.

Nadhled administrace redakčního systému Joomla! nalezneme v příloze B - strana 50.



Obr. 6 – Schéma MVC v Joomla! CMS

Zdroj: http://www.joomla.org/about-joomla.html.

5.3. Základní popis Dalénu CMS

Dalén je interní administrační systém navržený pro vývoj, zobrazování a úpravu webových aplikací vyvíjených firmou Miton. Vytvoření webové aplikace je umožněno složením různých typů obsahu, které jsou zastoupeny moduly. Nejpoužívanější typy modulů jsou například:

- Články
- Novinky
- Galerie
- Anketa
- Menu
- Katalog
- Diskuze / Komentáře

- Kontaktní formulář
- E-shop

Tyto moduly mohou být doplněny dalšími moduly se specifickými funkcemi pro konkrétní vyvíjený projekt, které lze doprogramovat.

Cíl projektu Dalén CMS je vytvořit nástroj, který bude vhodný jak pro vývoj náročných aplikací, tak bude obsahovat rozhraní pro rychlé postavení webu stavebnicovým způsobem.

Práce s Dalénem je řešena pomocí intuitivního programového klienta, který pracuje ve známém prostředí MS Windows a neliší se výrazně od běžných aplikací. Přes klienta lze velmi snadno spravovat celý obsah webové aplikace, měnit uspořádání stránek, přidávat nové, upravovat texty atd. Všechny změny se ihned projeví na výstupní prezentaci.

Nadhled administrace redakčního systému Dalén nalezneme v příloze C - strana 51.

6. Závěr

Při vývoji webové aplikace pomocí programovacího jazyka PHP, v počátků záleží na samotném programátorovi a na jeho dosavadních zkušenostech s programovacími technikami. Pokud víme, jakým způsobem funguje objektově orientované programování, není problém začít studovat jazyk PHP rovnou objektově orientovaným přístupem, a tudíž tak využít veškeré výhody verze PHP 5. Pokud však nemáme tak velké zkušenosti, je lepší si projít menším projektem právě jen ve strukturované verzi, za účelem pochopení syntaxe a struktury jazyka PHP. Vyvíjet webovou aplikaci čistě jen pomocí programovacího jazyka PHP, nevyužitím určitých systémů, je však vhodné maximálně pro tvorbu neprofesionální, méně rozsáhlé, spíše zkušební webové aplikace.

Framework využijeme pokud chceme vytvořit profesionální webovou aplikaci s rozsáhlými databázemi, jasným navrženým konceptem a složitou realizací. Jeho efektivnost je dána opětovnou využitelností již jednou programovaných funkcí, dodržováním určité architektury nasazením návrhového vzoru MVC, který je specifický právě pro PHP Frameworky. Důležitou a podstatnou roli při použití frameworku hraje fakt, že se na vyvíjené aplikaci podílí více členů. Každý se tedy specializuje na určitý problém vývoje a využívá nástroj framework jako efektivní vývojové prostředí.

CMS využijeme v případě, že máme již navržená řešení vyvíjené webové aplikace a konkrétní CMS nám je dokáže maximálně poskytnout, tudíž v případech, kdy webová aplikace nemá specifické, ale jen obecné funkce podporované právě určitým CMS. CMS lze však dále rozšiřovat v závislosti na tom, zda je CMS vytvořeno pro firemní účely nebo pro obecnou veřejnost. Pro firemní účely tento nástroj efektivně krátí čas, snižuje náklady spojené s vývojem a správou webové aplikace a lze ho neomezeně upravovat. Obecné veřejnosti poskytuje možnost vytvořit si a spravovat vlastní webovou aplikaci, aniž bychom museli být profesionální vývojáři, jsme však omezeni určitou funkcionalitou daného CMS.

Tuto funkcionalitu můžeme rozšířit pouze dalšími profesionálně naprogramovanými funkcemi, ale ne samotným přeprogramováním, pokud se ovšem řadíme mezi amatéry v této oblasti.

V mé bakalářské práci jsem se snažil popsat základní informace o výše zmiňovaných problematikách, jejich srovnání a pomoc pro začínající vývojáře, který z nástrojů je pro tvorbu webových aplikací efektivní v daný řešený problém.

Seznam použité literatury

Citace

[1] *Příloha A. Historie PHP a souvisejících projektů*. Dostupný z WWW: <http://php.mirror.camelnetwork.com/manual/cs/history.php>

[2] *PHP Frameworks*. Dostupný z WWW: < http://www.phpframeworks.com>

[3] *Zend Framework programmer´s reference guide* Dostupný z WWW: <http://framework.zend.com/manual/en>

[4] ALLEN ROB, LO NICK, BROWN STEVEN. *Zend Framework in Action*. Manning Publications. 2008. ISBN 1933988320

[5] BERNARD BOREK. *Úvod do architektury MVC*. Dostupný z WWW: <http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc>

[6] KOFLER M. , ÖGGL B.. *PHP 5 a MySQL 5 Průvodce webového programátora*. Computer Press. 2007. ISBN 8025118139

[7] MARKIEWICZ E. M. and LUCENA C. *Object Oriented Framework Development*. Dostupný z WWW: <http://www.acm.org/crossroads/xrds7-4/frameworks.html>

[8] ZICHA V. *Historie internetu*. Dostupný z WWW: <http://www.tvorba-webu.cz/php/historie.php>

Bibliografie

[9] *Joomla! Documentation*. Dostupný z WWW: <http://docs.joomla.org>

[10] *Zend Framework quick start*. Dostupný z WWW:
<http://framework.zend.com/docs/quickstart>

[11] VÁVRŮ V. *Úvod do Zend Frameworku*. Dostupný z WWW:
<http://vavru.cz/php/uvod-do-zend-frameworku>

[12] VÁVRŮ V. *Zend Framework - Hello World projekt (a MVC pattern).*
Dostupný z WWW:
<http://vavru.cz/php/zend-framework-hello-world-projekt-a-mvc-pattern>

[13] VÁVRŮ V. *Video PHP Seminář - Zend Framework.*
Dostupný z WWW:
<http://www.avc-cvut.cz/avc.php?id=5253>

Přílohy

Seznam příloh

Příloha A - příklady PHP frameworků

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ashLMVC	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell	-	✓	-	-	-	-	✓	-	-	✓	✓	✓	-
PhpOpenBiz	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-
WASP	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
ZooP	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

Příloha B - administrace redakčního systému Joomla!

Příloha C - administrace redakčního systému Dalén